UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

Peso: Segunda avaliação
Disciplina: Lógica e Algoritmo
Professor: Dr. Joaquim Assunção
Data de entrega: 23/set/2020, <21h
Data limite para a entrega: 28/Set/2020

Trabalho prático individual (*Regime especial (REDE)*)

Junte forças com outros programadores para criar um jogo (*SnakeX).* O jogo está parcialmente criado; interface e algumas funções estão prontas, bem como algumas variáveis estão definidas. Você tem a tarefa de criar algumas funções para que o jogo funcione.

O jogo roda em uma matriz `mat(151, 9c)`. Previamente preenchida com pontos ("**.**"). A cobra nada mais é do que o caractere 'C' sendo impresso na tela. A cada vez que 'C' aparece, o jogo pinta o caminho ilustrando o movimento que por padrão acontece a cada 1 segundo.

Você tem as seguintes tarefas:

- Preencha uma função chamada ***calculaCaminho***, que conta quantos **C**s foram colocados na tela.
    - Em ***calculaCaminho,*** imprima a saída na área reservada na tela.
- Preencha uma função chamada ***matKeyPress***, que recebe como parâmetro a tecla digitada (variável ***key***), e de acordo com a tecla, atualiza a variável global **dir.** Use as teclas **w, s, a, d**, para cima, baixo, esquerda, e direita, respectivamente.
- Preencha uma função ***tempoTimer*** que identifica a direção (**dir**) e atualiza a matriz com um '**C**'. Caso haja colisão, a direção é definida como 'N' e o jogo para mostrando *game over* (você só precisa definir a direção como 'N', o resto está pronto).
- Por fim**,** atualize o tempo de acordo com a pontuação. Multiplique por 0.8 caso 10 pontos; 0.7 caso 20 pontos; 0.6 caso 30 pontos; e 0.5 caso 50 pontos.

Variáveis globais já definidas

`tela` //Nome da interface

`mat` // matriz na tela, use ***mat.Cells[col, lin]*** para alterar os valores de uma célula

`tlinhas` : integer; //Total de linhas

`tcolunas` : integer; //Total de colunas

`lin, col` : integer; //linha e coluna atual

`dir` : char; //direção atual (U, D, L, R, N)

`tela.tam_caminho.Caption` //recebe uma *string* e imprime na tela

`tempo` : integer; //Controla a velocidade da cobra, use ***Interval*** para ajustar

Exemplo de declaração de Variáveis

```
var
    c,l : integer;
```

Funções

`intToStr()` //retorna uma string dado um inteiro

`round()` //Converte um float para um integer

Sintaxe

SE … ENTAO → `if( )then` //precisa () para sub-testes lógicos e () para agregar todos

{ … } → `begin ... end`

== → `=`

<- ou = → `:=`

Para var de ini ate fim { ... } → **`for` *`var`* `:=` *`ini`* `to` *`fim`* `do begin ... end`**

Strings e caracteres usam aspas simples.

Avaliação

O trabalho perfeito é entregue, compilável, em no máximo 2 dias (23/set, <21h). Pequenos erros são corrigidos e descontados (de 0.1 a 0.3 dependendo do erro). Erros maiores geram uma nova tentativa. Você tem 3 tentativas! Cada tentativa errada, desconta 0.5 pontos. Cada dia após dia 23, desconta 0.4 pontos.

Exemplos:

- Trabalho compilável entregue dia 24, na segunda tentativa → 9.1 pontos.

- Trabalho compilável entregue dia 25, na terceira tentativa → 8.2 pontos.

- Trabalho compilável entregue na terceira tentativa, no dia 28 → 7 pontos.

- Trabalho não compilável, serão avaliadas as funções corretas → <7 pontos.